Quarkus Logging

JEE Microservices

@ CGS IT – 2023

Version 1.0.5

Inhalt

- Quarkus – Supported Logging APIS
- Quarkus/Jboss – Log Levels
- Quarkus - Runtime configuration
- Quarkus - Log Handlers
- Quarkus – Dokumentation Links

Quarkus – Supported Logging APIS

- JBoss Logging
- JDK java.util.logging (also called JUL)
- SLF4J
- Apache Commons Logging

Internally Quarkus uses JBoss Logging; you can also use it inside your application so that no other dependencies should be added for your logs.

Quarkus/Jboss – Log Levels

OFF	Special level to turn off logging.
FATAL	A critical service failure/complete inability to service requests of any kind.
ERROR	A significant disruption in a request or the inability to service a request.
WARN	A non-critical service error or problem that may not require immediate correction.
INFO	Service lifecycle events or important related very-low-frequency information.
DEBUG	Messages that convey extra information regarding lifecycle or non-request-bound events which may be helpful for debugging.
TRACE	Messages that convey extra per-request debugging information that may be very high frequency.
ALL	Special level for all messages including custom levels.

Logging Example

- Jboss logger
- LOG.infov Message Format

```java
import org.jboss.logging.Logger;

/**
 * doku
 */
@Path("/helloDemo")
public class DemoResource {
    private static final Logger LOG = Logger.getLogger(DemoResource.class);

    /**
     * http://localhost:8080/helloDemo/echo/<inputString>
     * @param inputString
     * @return service processed input string
     */
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    @Path("/createCMWithName/{inputString}")
    public String createCMWithName(@PathParam("inputString") String inputString) {
        LOG.infov("log: {0}", inputString);
        return "Hello [" + service.createChatMessageDBAndReturnCount(inputString ) + "]";
    }
```

```
--  ----  --  -----   ---  -- ----  ------
 --/ __ \/ / / / _ | / _ \/ //_/ / / / __/
 -/ /_/ / /_/ / __ |/ , _/ ,< / /_/ /\ \
--\___\_\____/_/ |_/_/|_/_/|_|\____/___/
2023-02-20 22:19:43,152 INFO  [io.quarkus] (Quarkus Main Thread) demo3 1.0-SNAPSHOT on JVM (powered by Quarkus 2.16.3.Final) started in 0.412s. Listening on: http://localhost:8080
2023-02-20 22:19:43,153 INFO  [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
2023-02-20 22:19:43,153 INFO  [io.quarkus] (Quarkus Main Thread) Installed features: [agroal, cdi, hibernate-orm, jdbc-postgresql, narayana-jta, resteasy, resteasy-jackson,
 smallrye-context-propagation, smallrye-openapi, swagger-ui, vertx]
2023-02-20 22:19:43,154 INFO  [io.qua.dep.dev.RuntimeUpdatesProcessor] (vert.x-worker-thread-0) Live reload total time: 0.612s
2023-02-20 22:19:43,156 INFO  [com.exa.DemoResource] (executor-thread-0) log: Chris
```

CGS ITSolutions

Injecting a Logger

- You can also inject a configured org.jboss.logging.Logger instance in your beans and resource classes.
- 1. The FQCN of the declaring class is used as a logger name, will be used.
  org.jboss.logging.Logger.getLogger(SimpleBean.class)
- 2. In this case, the name foo is used as a logger name, i.e. org.jboss.logging.Logger.getLogger("foo") will be used.

```
import org.jboss.logging.Logger;

@ApplicationScoped
class SimpleBean {

   @Inject
   Logger log; 1

   @LoggerName("foo")
   Logger fooLog; 2

   public void ping() {
     log.info("Simple!");
     fooLog.info("Goes to _foo_ logger!");
   }
}
```

CGS ITSolutions

Quarkus - Runtime configuration

Procedure

- Configure the logging in your `application.properties` file:

  The following example shows how to set the default logging level to `INFO` logging and include Hibernate `DEBUG` logs:

  src/main/resources/application.properties

```
quarkus.log.level=INFO
quarkus.log.category."org.hibernate".level=DEBUG
```

Quarkus Logging Configuration Reference:
https://quarkus.io/guides/logging#loggingConfigurationReference

Quarkus - Log Handlers

The console log handler is enabled by default. It outputs all log events to the console of your application (typically to the system's stdout).

For details of its configuration options, see the Console Logging configuration reference.

```
quarkus.log.file.enable=true # Send output to a trace.log file under the /tmp directory
quarkus.log.file.path=quarkus-log.log
quarkus.log.file.level=debug
quarkus.log.file.format=%d{HH:mm:ss} %-5p [%c{2.}] (%t) %s%e%n
```

Quarkus – Dokumentation Links

- Quarkus Logging

https://quarkus.io/guides/logging

- Jboss Logging
  https://docs.jboss.org/seam/3/latest/reference/en-US/html/solder-logging.html

Danke für Ihre Aufmerksamkeit

CGS ITSolutions